性能

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 用户，第三方系统 |
| 刺激 | 非周期性的数据请求，操作请求 |
| 制品 | Broker模块，服务器模块，request bus |
| 环境 | 系统正常运行时，系统高负载时 |
| 响应 | 系统执行用户请求的操作；处理用户数据；返回数据 |
| 响应度量 | 请求响应延迟时间不超过5s  请求丢失的比例不超过0.001% |

#### 选择元素

选择的元素是安全模块，包括broker部分的authorityCheck，位于broker和 server之间的security Check。

#### 选择ASR

第二次迭代选择的ASR是安全性。这个模块主要负责系统的安全控制。

#### 策略表和决策

|  |  |  |  |
| --- | --- | --- | --- |
|  | 优点 | 缺点 | 是否采用 |
| 检测请求模式 | 通过检查正常服务请求的模式和机器指令，来控制访问权限。避免抢票插件不正常的操作数据。避免刷票程序占据带宽，增强性能。 | 增加系统复杂度，需要专门的模块来记录请求模式，比较请求模式。可能增加每一条请求的时间，降低性能。 | 采用 |
| 检测服务拒绝情况 | 通过检查同一身份认证多次失败的情况，能够防止恶意的测试用户密码。 | 同上 | 采用 |
| 识别信息一致性 | 通过检验hash值或者checksum来确定数据一致性。能够避免用户数据直接传输或者存储，在数据泄露时造成重大损失。 | 增加信息传输总量，占有带宽，降低性能。 | 采用 |
| 识别角色 | 对外部输入系统的资源加以识别，防止脚本注入攻击等。 | 增加系统复杂度。 | 采用 |
| 用户认证 | 通过识别用户身份，避免不当的数据操作，其他用户修改了某个用户的个人信息。 | 需要与其他组件交互，存储用户数据，增加系统复杂度。 | 采用 |
| 用户授权 | 通过给用户授予合适的权限，限制用户的能力，避免用户对重要数据的不当操作。 | 无 | 采用 |
| 限制对计算机资源的访问 | 通过限制直接访问计算机资源，避免暴露系统内部细节。减少了黑客攻击的可能。 | 无 | 采用 |
| 最小化系统的攻击面 | 通过减少系统访问面，要求所有的请求必须经过安全处理模块，能够集中处理所有的攻击。 | 因为所有的请求都必须经过安全检查部分，所以该组件一旦遭到攻击失效，会影响系统的可用性安全性。 | 采用 |
| 当攻击发生时删除敏感的资源 | 系统已经遭到潜在的攻击，与其使重要数据被窃取，不如直接销毁，减小损失的影响。 | 对于不是特别敏感的资源，删除导致的损失过大 | 不采用，该系统没有敏感资源 |